International Journal of Research in Computer Science
eISSN 2249-8265 Volume 3 Issue 2 (2013) pp. 11-21
www.ijorcs.org, A Unit of White Globe Publications
doi: 10.7815/ijorcs. 32.2013.061

AN ENHANCED FRAMEWORK FOR IMPROVING SPATIO-TEMPORAL QUERIES FOR GLOBAL POSITIONING SYSTEMS

Ishraq Al-Fataftah[1], Jalal Atoum[2]

[1 2]*Princess Sumaya University for Technology, Amman, JORDAN*
Email: [1]ishraqabd85@hotmail.com, [2]atoum@psut.edu.jo

Abstract: *To efficiently process continuous spatio-temporal queries, we need to efficiently and effectively handle large number of moving objects and continuous updates on these queries. In this paper, we propose a framework that employs a new indexing algorithm that is built on top of SQL Server 2008 and avoid the overhead related to R-Tree indexing. To answer range queries, we utilize dynamic materialized view concept to efficiently handle update queries. We propose an adaptive safe region to reduce communication costs between the client and the server and to minimize position update load. Caching of results was utilized to enhance the overall performance of the framework. To handle concurrent spatio-temporal queries, we utilize publish/subscribe paradigm to group similar queries and efficiently process these requests. Experiments show that the overall proposed framework performance was able to outperform R-Tree index and produce promising and satisfactory results.*

Keywords: *Enhanced Framework, GPS, Spatio-Temporal Queries, SQL Server.*

I. INTRODUCTION

The ascendance of mobile devices and advances in wireless technology nowadays has layered the landscape for the spread of services provided based on the geo-location of users. Whereas location based services are not new, the early adaptors in the 90's would have no clue on how important location data would be and the impact it imposes in our daily life. Location based services (LBS) are services that provide geographic information and geo-processing power to mobile users based on their current locations or to stationary (non-moving) users according to the stationary object/mobile of their interest [5].

The emergence of positioning technologies especially global positioning systems (GPS) have revolutionized the mobile market and had a great impact in enhancing location based services. The level of accuracy produced by GPS technology has encouraged different players in the mobile market to invest in GPS capabilities whether as built-in technology in smart phones by mobile manufacturers or as an added value by mobile operators and service providers. Social networking tools such as Facebook, Twitter, Foursquare and Google+ also have helped in spreading LBS and in helping users to recognize the benefit gained from such services when compared to traditional ones.

LBS has been developed from traditional navigation and tracking systems to advanced social services, safety and security services, payment and billing, advertisements, gaming and entertainment services. Navigation systems nowadays not only can show you the route from place A to place B but also will walk you through the desired route turn by turn showing the exact distance and time left to reach the exact place. Novel search capabilities such as finding the location of your friends, nearest gas station and finding public libraries in the route you are driving in. Modified search options based on user locations have minimized the search domain to provide more precise and useful information to users upon request.

Moreover, context aware advertisements tailored based on user location, speed, direction or past history are gaining more success since these advertisements are becoming more useful and helpful for users [6]. These are some examples on how LBS can facilitate and ease our lives. Statistics are showing that 35% from mobiles around the world are smart phones, 47% from all phones will be smart phones by 2015, 95% from smart phones users are looking for location data, LBS users reached 63 million in 2009 and expected to grow in 2015 and reach about 468 million users [9]. Such statistics indicate the increased awareness toward LBS.

Although studies are showing that smart phone users are not yet reaching the critical mass neither those using LBS, the importance of data in the digital world we are living in is increasing and turning into the number one commodity in this world. Data provided in terms of LBS are continuously changing in time and location and the need to provide the right data in the right time and location is being critical for such services. For instance, if searching for the nearest hospital in times of emergency are not delivered in

near real time and with high degree of accuracy will put lives into danger and turn these services into an unreliable and an inefficient facilities. This type of queries are called spatio-temporal queries in which objects change their location and shape overtime and data can be described in both space dimension and time dimension [4]. Any delay in these queries will yield an invalid or obsolete result.

The contribution of this paper is as follows:

- We propose a framework that better utilizes location based services resources in highly dynamic environments and supports dynamic queries for moving objects.
- We propose an algorithm for spatial indexing that enhances the performance and response time of spatio-temporal queries.
- We utilize safe region concept by proposing an adaptive safe region that helps in minimizing communication between client and server side.
- We provide a comprehensive set of experiments that show the performance of the overall framework and how it is more efficient than just using R-tree indexing in SQL Server 2008.

The rest of the paper is organized as follows: Section 2 highlights main related work for handling processing of spatio-temporal queries. Section 3 demonstrates a quick overview about spatio-temporal queries. Section 4 proposes a new framework for handling range queries. Section 5 provides an extensive set of experiments to assess the performance of the proposed framework. Finally, section 6 concludes the paper and presents directions for future work.

II. RELATED WORK

Spatio-temporal databases have been the subject of various researches. For instance the *CHOROCHRONOS* is a European research project that studied spatio-temporal databases. Advances in several areas have been accomplished including query processing, data modeling, indexing and architecture for *STDBMSs* [7].

A lot of research work had focused on one aspect of these queries such as the work that had focused on moving queries over stationary objects, other work had focused on stationary range queries [8,10]. In addition, these researches had concentrated on the validity of query results when being processed, meaning that query results are considered valid only if it is within a certain region such as valid region concept [8] or within a certain period of time such as valid time [10]. Moreover, to overcome the problem of processing capabilities of clients, caching techniques have been developed either in the server side or client side. If the expected location of moving objects is known in advance by calculating the velocity of moving objects, data can be pre-computed [11].

PLACE, Pervasive Location-Aware Computing Environment is a framework for processing both moving queries and moving objects in addition to keeping track of the history of these objects [1] and [3]. What distinguishes *PLACE* from the other works discussed above, is its ability to process different types of continuous spatio-temporal queries, the use of incremental updates to evaluate new results, scalability to support concurrent execution of queries through sharing the execution of similar queries by grouping them based on certain criteria and no processing is to be performed on the client side. Yaun and others [12] had suggested a framework for handling past and present/future position of objects in the disk storage and the memory separately. By using a PCFI and PCFI + index structure for tracking moving objects, past and present/future queries are processed efficiently in disk and memory.

Mokbel and others [13] had proposed an algorithm called SINA: Scalable Incremental Hashed based algorithm to process concurrent spatio-temporal queries. The purpose of this algorithm is to efficiently process a large number of concurrent queries using an incremental evaluation. Scalability was achieved by using a shared execution paradigm by a spatial join between moving objects and moving queries.

III. SPATIO-TEMPORAL QUERIES

Spatio-temporal queries are queries in which objects change their location and shape overtime in which data can be described in both space dimension and time dimension [1,3,4]. These queries are a variation from spatial queries and temporal queries. Spatial queries are queries that are initiated based on the position or location of objects, while temporal queries are queries initiated in temporal databases in which we are interested only in the current state of the object in which we only store the current state of objects not past states.

The main spatial queries that fall under spatio-temporal queries are:

A. Window Queries (Range)

These types of queries retrieve data that intersects a certain window or region. R-Tree is an example on this type of queries as shown in *Figure 1*. In this case, we are interested only in objects that reside within the boundaries of the query window; any object that is outside this window will not be retrieved. Continuous range queries are queries that remain active over a period of time and are continuously evaluated to retrieve accurate answers (Stojanovic and Dordevic-Kajan, 2003) [5].

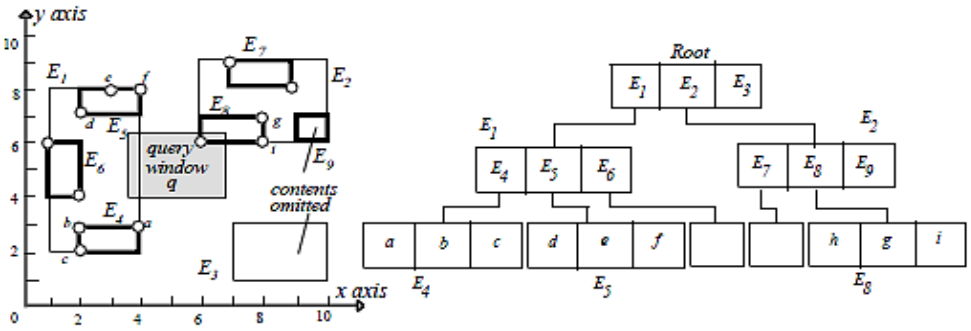

Figure 1: Window query using R-Tree

B. Nearest neighbor queries (NN):

This type of queries retrieve objects that are near the current location of the mobile device, as shown in *Figure 2*. In such scenario, we can cache the result of this query at the client side as long as the mobile device did not change its location. But mobile devices are moving all the time, to overcome this issue the concept of valid region was introduced. When a mobile device issues NN query, the result returned include with it the region in which this result is considered valid and the mobile determines whether to issue new request or use the valid region by determining whether it is still in this region or it moved outside [11].

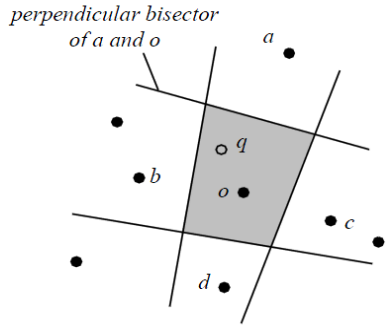

Figure 2: KNN Query

C. Reverse nearest neighbor queries:

It is a variation from NN queries in which it works in the reverse order. These types of queries are interested in finding all objects in which *q* is their nearest neighbor.

D. Historical queries:

These queries are interested in past location of objects. This requires the system to store the location of objects in different period of time. Usually, when the object changes its location, a record is moved to the history database and calculated from that database.

E. Now queries:

These queries are interested in the current location of objects. This requires the system to maintain a record for the latest position of all objects.

F. Future queries:

These queries are interested in predicting the future location of the objects. Extra information such as the velocity of moving objects should be stored along with the location of each object.

This paper handles only NOW queries in Range spatio-temporal queries.

IV. THE FRAMEWORK

The proposed framework is built around a publish/subscribe paradigm in which moving objects and queries are mapped together through a notification system/engine. *Figure 3* demonstrates the infra-structure for the framework proposed. The reason for choosing this approach is that many suggested solutions for handling the large number of concurrent queries is through sharing where similar queries are grouped together based on rules extracted from each query and same answer are returned for different query owners such in [13, 14 ,1]. Special engines were developed to carry out these operations. The publish/subscribe paradigm simplifies this concept in which similar queries are treated as subscriptions, these subscriptions share the same interests, once a data is available (in our case upon requesting of data), the notification engine is responsible for delivering the results to all subscribers. The main component of the framework is the Notification system/engine that is responsible for collecting location data from publishers and subscribers, handling and processing continuous queries and returning results to the subscribers.

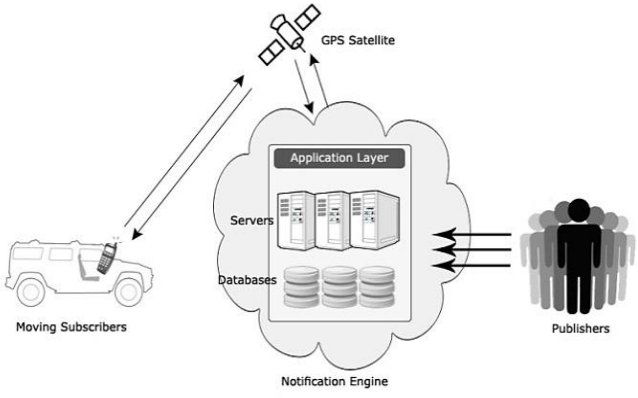

Figure 3: Main Framework Infrastructure

Let us consider an example in which we are interested in retrieving all restaurants within one kilometer radius. In this scenario, mobile devices as moving objects (when driving a car or walking) are interested in searching for static objects (Restaurants) within a range of 1K radius. This type of spatio-temporal queries is of type moving queries over static objects in which we have built our framework around. The client application installed on the mobile device will provide the client with a graphical user interface (GUI) that will display a geographical map such as Google maps. An indicator of the client current location will be displayed on the map that will fit the display resolution of the mobile device. The application will request the service providing the client spatial coordinates obtained from the GPS embedded within the mobile device.

Publishers in this framework are static objects we are interested in providing their data so that others can search for them. For instance, restaurants in our example are publishers which register their location and type of service provided to the notification system.

When a mobile device moves, it obtains its location through GPS system embedded within the device and transmits its location through a wireless communication system to the notification engine. The efficiency of the system is highly dependent on the wireless communication network that affects the transmission and reception of information. We will disregard any implication of the wireless communication network in this framework. The publishers would provide information about their services and location information through traditional means of registration.

The framework works in the following way:

1. When a mobile device issues a request for a service, a subscription ticket is issued to identify the mobile object and its current location.
2. A circular safe region is constructed around the moving mobile device to minimize the number of location updates issued by the mobile device.
3. The notification engine will process the query based on the location provided.
4. The result will be published back to the mobile device.
5. A cached version of the results will be stored to answer any query issued and have the same query results.

The main idea of using a safe region is that as long as the mobile device is within the safe region, there is no need to continuously issue update to the mobile device since the result will be the same. The reason for using a circular safe region is that the underlying Euclidian space in this framework is partitioned as circular regions and the indexing structure proposed also is built over circular regions. Moreover, research papers for using the shape of safe region proved that using circular safe region has more advantages over using rectangular safe region [15]. *Figure 4* depicts the circular safe region. As long a moving object *(O)* is within the safe region *(O_1-O_k)*, no query will be issued although the location of the object is changing. Once an object is outside the safe region *(O_{k+1})*, an update location query would be issued to retrieve an updated set of results.

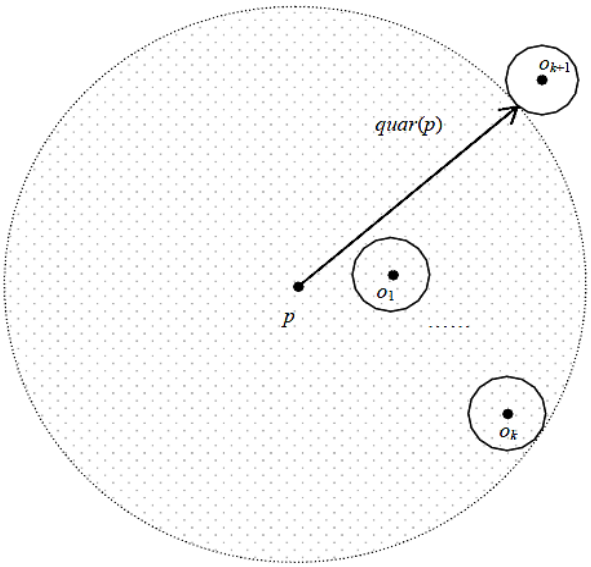

Figure 4: Circular Safe Region

The caching manager is responsible for caching the query result for each query region to process any query issued by other objects that share the same query region and type of service. The notification engine is responsible for handling concurrent queries by grouping similar objects that share the same safe region and type of service (Subscription model).

As we have mentioned earlier, the notification engine is the core part of the system where all retrieval and creation operations are taken place. The design decision taken for searching and retrieving spatial data is done through the use of a spatial indexing structure over the publisher table. Our testing on actual data showed a significant variation in the query response time when data retrieved from tables is directly compared to data retrieved with the presence of a spatial index which was an R-Tree index. Therefore, we have proposed an algorithm that partition the spatial space using a grid into equally spaced partitions, each partition is covered with a circular shape to construct the structure of the index as shown in *Figure 5*. For the uncovered areas in the intersection point between the four squares we use a filter circle to include these points in the index. Each circle will cover a number of points that lies on its circumference and inside the circle; these points will be used as parent and child nodes for constructing the index tree.

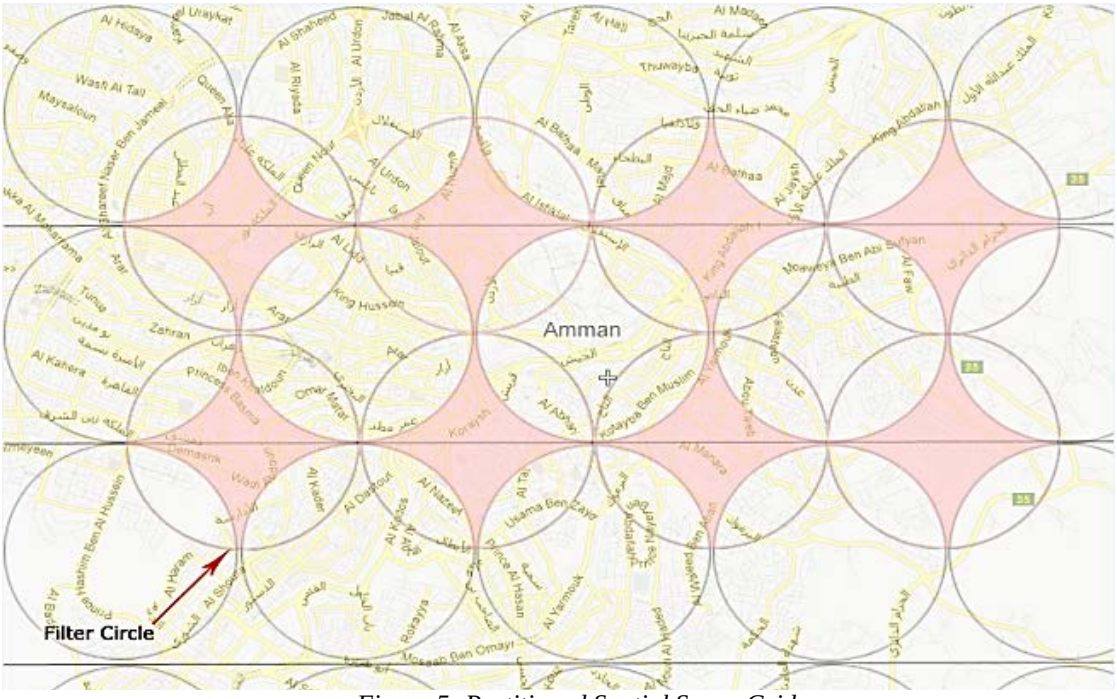

Figure 5: Partitioned Spatial Space Grid

Spatial indexes are considered heavy and impose a huge overhead for insertion and deletion operations. As spatio-temporal queries are continuously changing their locations, new publishers are added to the system and old publishers may be deleted from the system, we have decided to take advantage of materialized views to represent portion of data subscribers interested in rather than querying over the original tables that maintains a huge amount of data. To overcome the maintainability issues for the MV, we have decided to use a dynamic materialized view based on work in [16], this view is controlled by a control table that holds the geo-location of the current subscribers (area of interest), so that only updates to the underlying tables that affect this area will be materialized.

Caching manager is responsible for caching results to a cache table to answer any query that share the same subscription interest and handle out of sync cases. Caching of data is triggered whenever the moving object reaches half the range of the interests region in order to pre-compute data and provide more precise results to subscribers in a timely manner. *Figure 6* summarizes the proposed framework architecture design.

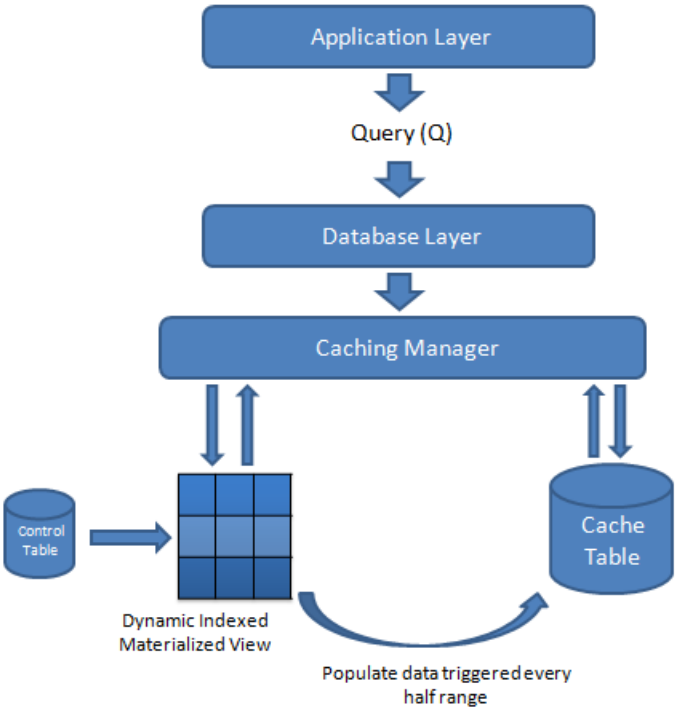

Figure 6: Proposed Framework Architecture

A. Proposed Indexing Algorithm

The main algorithm proposed to index the spatial data is presented below. In a nutshell, this algorithm divides the spatial space into equally spaced portions, each with an overlay circle shape that set the main structure of the algorithm. For each circle, we consider points that lie on the circle circumference as parent nodes and points that lie within the circle as child nodes. The tree will hold parent nodes with children that lie on the left side of the circle as left sub tree and children that lie on the right side of the circle as right sub tree. When the root node is created, it is placed in the first page. All parents then will be inserted in the same page until the pages is full. New page will be created to hold subsequent nodes.

When a point is involved in an insert, delete or search operation, parent nodes that intersects the point of interest or those lie within its range are retrieved. All children that satisfy the same criteria will be retrieved. The search relies on the location of the node in the left or right side of the circle in order to filter and minimize nodes to search through.

```
// Loop through all cells in the spatial space grid and filter circles
Foreach(cell in space grid)
{
        // Retrieve only data that intersects the circle cell
        Get all data that intersects cell into TempIntersectTable;
        Foreach(point in TempIntersectTable)
        {
                // Select points that lies on circle circumference as parent nodes
                IF(point lies on cell circumference)
                {
                        // Parent node lies on the Left side of the cell
                        IF(point.distance <= cell center)
                        {
                                Insert record into ParentTable with LeftChild = 'L' AND RightChild =
                                NULL;
                        }
                        // Parent node lies on the Right side of the cell
                        ELSE
                        {
                                Insert record into ParentTable with LeftChild = NULL AND RightChild =
                                'R';
                        }
                }
                // Select points that lies inside circle as child node
                ELSE
                {
                        // Child node lies on the Left side of the cell
                        IF(point.distance <= cell center)
                        {
                                ChooseParent('L');
                                Insert record into ChildrenTable with Left = 'L' AND Right = NULL;
                        }
                        // Child node lies on the Right side of the cell
                        ELSE
                        {
                                ChooseParent('R');
                                Insert record into ChildrenTable with LeftChild = NULL AND RightChild
                                = 'R';
                        }
                }
        }
}
// Select a parent node from the parents table (nodes that lies on circle circumference)
// first node that is not full and lies on left or right side of the circle
// If no such node exists, create a dummy node that will be the parent of all nodes
Function ChooseParent(location)
{
        var parentNode;
        SELECT parentNode = TOP record FROM ParentTable
```

```
                WHERE record is not full AND
                (LeftChild = location OR RightChild = location);
                // If such node exists, then return parentNode
                IF(parentNode !=null)
                {
                        Return parentNode;
                }
                // If no such node exists, create dummy node
                ELSE
                {
                        var dummyParentNode = sequential number AND LeftChild='L' AND RightChild='R';
                        Return dummyParentNode;
                }
}
```

B. Dynamic Materialized View

For retrieving only portion of data that each subscriber is interested in and lies within the specified range, a dynamic materialized view will be created. The materialized view will be created over the indexed publishers table and will retrieve data that only exists within the control table. For instance, if a subscriber is within Area (A1) and requested all data within range (R), this information will be located in the control table. When the query is processed, the MV will return only locations that lie within R range or distance from A1 instead of searching the entire publishers table. The script below demonstrates how to create dynamic materialized view.

```
// Create Control Table
CREATE TABLE [dbo].[Control](
      [ObjectID] [int] IDENTITY(1,1) NOT NULL,
      [Time] [datetime] NOT NULL,
      [GLocation] [geography] NOT NULL

// Create Dynamic Materialized View
CREATE VIEW V_SpatialView WITH SCHEMABINDING AS
SELECT P.locId,
         P.city,
         P.country,
         P.postalCode,
         P.region
FROM dbo.Publishers P
WHERE EXISTS
            (SELECT 1
             FROM dbo.[Control] ctrl
             WHERE ctrl.GLocation.STDistance(P.GEOG) <=@Range)
```

The control table used in this design is of type Range Control table. This table supports range predicates for range or point queries such as calculating distance from the point *(STDistance)*, determine within a point within specific area *(STWithin)*, retrieving all points that intersects a certain point *(STIntersects)*, etc.

Using dynamic materialized views inherits the following advantages:

1. Only data that satisfies the client request will be created as an actual table stored in the database to enhance query processing and data retrieval.
2. Spatial Index will be created against the MV and will be efficiently utilized against partial set of data instead of entire spatial table.
3. Continuous maintenance operations as a result of introducing new publishers or removing existing ones are efficiently materialized resulting in enhanced maintenance and re-indexing operations.

C. Caching Manager

When a query is made against the notification engine, the caching manager check the cache table for an answer for this query, if the cache is empty indicating this is the first request by the object, the cache manager will answer the query from the dynamic indexed materialized view. A record of the new object will be inserted in the control table and the query is retrieved from the MV. After the answer is returned to the object, the caching manager populates the cache table with the answer of the issued query. This will ensure that subsequent requests made from the subscribers to the same service and same safe

region will be answered directly from the cached version of the results.

Moreover, when the moving object reach half the distance of the range specified in the search query, the caching manager will be triggered to issue an automatic query that updates the control table with a new location, execute the query against the MV and populate the cache table with new data. This will ensure that search results will automatically be executed and available with new and more precise data, the client will need only to issue the first request and send an indication that the object has reached half the way and there is no need for the client to issue subsequent update requests to maintain data.

V. PERFORMANCE EVALUATION

In this section, we will present an evaluation of the performance of the proposed framework to assess its efficiency in comparison with R-Tree indexing approach implemented in SQL Server. We have created a simulation environment for testing the proposed framework using SQL Server 2008 R2. We ran a series of experiments to evaluate each component of the framework as a single unit.

A. Preliminaries

All experiments were performed on a workstation with 2.53 GHz Core 2 Duo processor, 4 GB memory and one 500 GB disk running Windows 7 operating system. We have used for testing real sample spatial database for world cities and countries that is downloaded from (MaxMind).

The performance of queries response time was measured using the following properties:

1. Dataset size: the number of objects that lie within a specific range to determine the speed of indexing operations and data retrieval and transfer rate. In addition, the maintenance operations on the materialized view are highly dependent on data size. We have considered data sizes within a range between 300,000 and 1,300,000 objects.
2. Range/Window size: this property has a great impact on the number of objects covered by each node in the indexing structure and on how much data can be returned. We have considered window size of 5 and 10 Kilometers.
3. Type of query: we have considered Range spatio-temporal queries that are concerned with NOW queries.
4. Number of concurrent queries: The publish/ subscribe paradigm and caching is used to handle multiple concurrent queries over the same area of interest.

Moreover, the client machine physical properties including CPU speed, memory size and other properties have a direct impact on the query processing operations including search, indexing operations and caching. This should be kept in mind since better physical properties provides better results.

B. Testing Results

We conducted a number of experiments to measure the performance of the proposed solution to enhance range queries in spatio-temporal database. We will first evaluate the proposed spatial index algorithm we have developed in this framework for search operations.

We ran the test over different sample of data series (300,000, 1,000,000 and 1,300,3000 records) one hundred consecutive times. As shown in *Figure 7*, the time taken to execute the queries measured in milliseconds (ms) shows an acceptable execution time even for large data size. In addition, the algorithm maintained a stable behavior when executed repeatedly overtime; the same results were obtained when we ran the algorithm more than a hundred times.

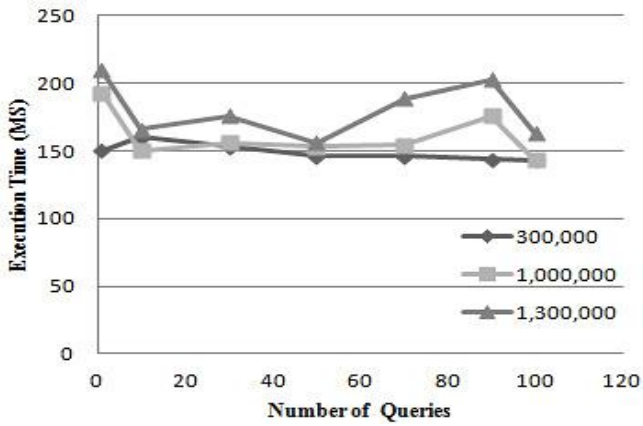

Figure 7: Proposed Index CPU Time (Search)

To assess the efficiency of our proposed algorithm, we have compared its performance against a spatial table with the application of R-Tree index as shown in *Figure 8*. It can be shown that a significant improvement have been obtained when compared to a table with no index. The reason is that, applying an index on the table where data stored in a well-defined data structure significantly improved the data retrieval. Moreover, storing the data in a tree based on a circular shape where points are classified as parent and child nodes based on their location on the circumference or inside a filter circle enhanced data storage in physical pages and data retrieval CPU time. Also, it can be noticed that the performance of the R-Tree is not stable when executed repeatedly over time in contrast to our proposed algorithm that maintained a stable behavior when executed repeatedly over time. This is due to the points were grouped based on a spatial geometry that enclosed points within a specific region in a well-defined tree and sub-tree where no overhead for handling overlapped regions is necessary.

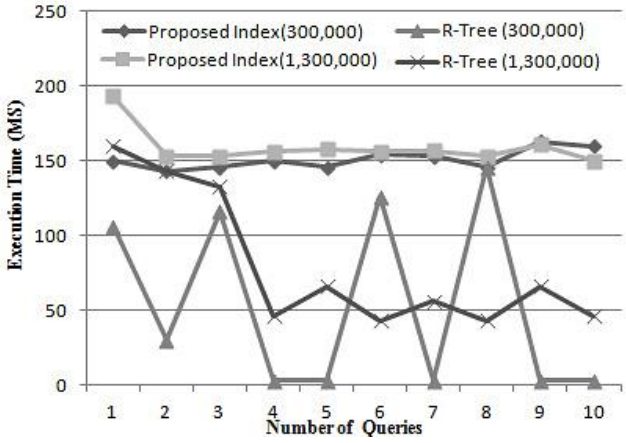

Figure 8: Proposed Index CPU Time

We ran a series of experiments to measure the performance of insertion and deletion operations using the proposed indexing algorithm in *Figures 9 and 10*, respectively. We can notice that these operations does not produce any overhead since the resulting tree does not need to be balanced so no additional computation is needed. In addition, the simplicity in selecting parent nodes and creating new sub-trees gave our indexing algorithm advantage over R-Tree index especially in the deletion operation.

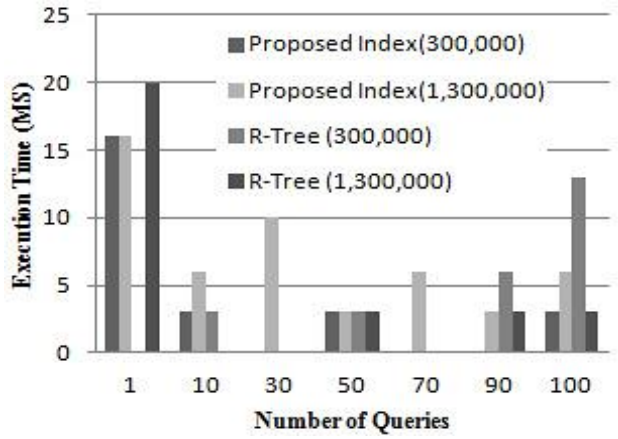

Figure 9: CPU Time (Insert)

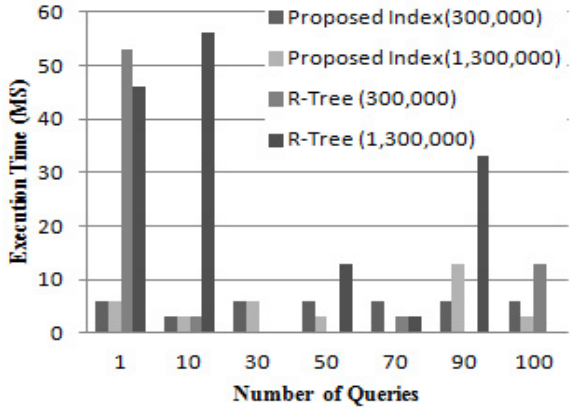

Figure 10: CPU Time (Delete)

To demonstrate the effectiveness of using dynamic materialized view, we have compared the behavior of the dynamic MV against a fully MV and a regular view. Materialized views reside in disk where views are generated on the fly in main memory. The execution plan of the three scenarios will show that fully materialized views are shown as single lookup whereas dynamic materialized view will be shown with two branches one for the control table and the other for the underlying table.

Figure 11 depicts the advantage gained for using a dynamic materialized view over fully materialized view and regular view when executed several times for different location requests. Using a control table that contains the current location of the moving objects and creating a materialized view that calculates the range query based on the current location of the moving object produce more enhanced results even when the location of the moving object was continuously updated.

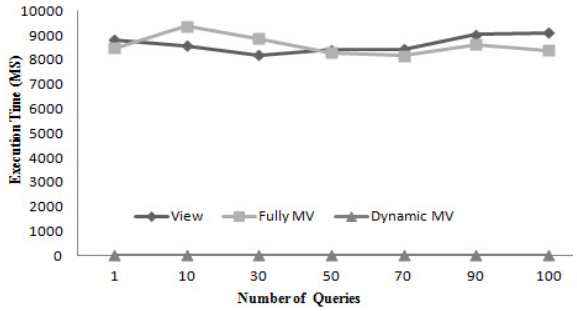

Figure 11: Search query response time using different views

Figure 12 demonstrates the performance of the framework from issuing the first request through the entire process compared with R-Tree index. We can notice that both methods register the same performance for the first request then we can notice how the continuous change in the moving object location affected the R-Tree index performance where the response time increased overtime whereas our proposed framework outperformed the R-Tree index where we noticed an enhanced response time overtime which is the purpose of this thesis.

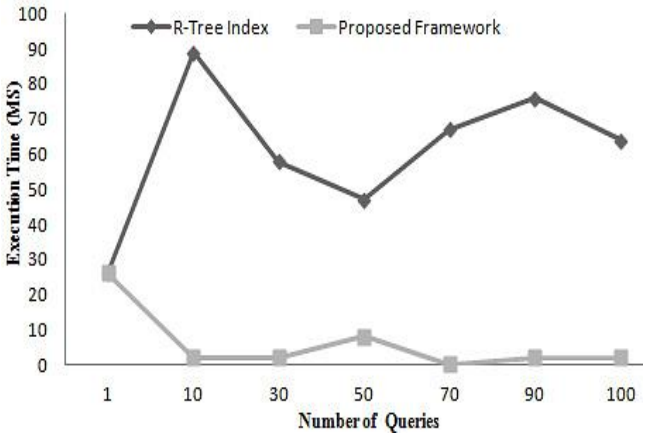

Figure 12: Proposed Framework CPU Time

Finally, we have measured the I/O statistics for both the R-Tree index and our proposed framework.

Table 1 demonstrates I/O statistics obtained from both operations including average and total I/O operations (average and total number of IO operations), average and total logical reads (average and total number of pages read from data cache), average and total logical writes (average and total number of pages written to cache data), average physical reads (average number of pages read from disk) and total physical writes (total number of pages written to disk). The proposed algorithm outperformed the R-Tree index in terms of I/O operations as indicated in *Figure 13* as well.

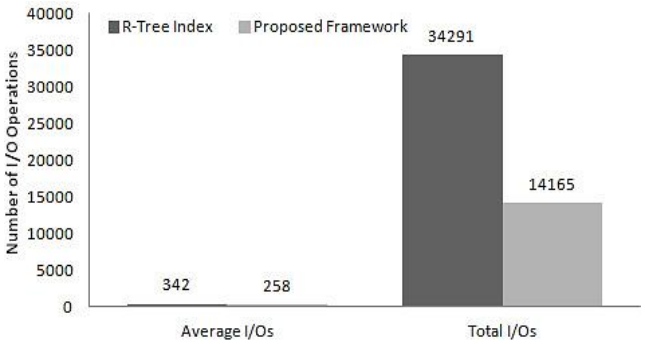

Figure 13: Proposed framework and R-Tree I/O operations.

Table 1: I/O operations for R-Tree Index and the Proposed Framework

	Count	Avg I/Os	Total I/Os	Avg Logical Reads	Total Logical Reads	Avg Logical Writes	Total Logical Writes	Avg Physical Reads	Total Physical Reads
R-Tree Index	100	342	34291	342	34291	0	0	0	0
Proposed Framework	100	258	14165	256	14163	2	2	0	0

VI. CONCLUSION AND FUTURE WORK

This paper had addressed the challenge of handling range queries for moving queries over static objects in location based services. We proposed a framework that addressed the need to efficiently and effectively answer continuous queries for concurrent users at the same time. Concurrent query execution is handled by utilizing the publish/subscribe paradigm. Moreover, an adaptive safe region is introduced to reduce the overhead of issuing continuous updates to the location of the moving object. Efficient spatial index over dynamic materialized views is proposed to reduce the response time for evaluating the continuous queries. A caching manager is proposed to handle update requests from the safe region and answer all subsequent requests from concurrent users. This approach had outperformed the R-Tree index when executed several times over time in terms of CPU performance time and I/O operations which had enhanced the overall response time for spatio-temporal queries.

There is a number of promising directions we can extend for future work based on research work presented in this paper. For the proposed indexing structure, it was built on the concept of range queries that is directly dependent on the window/range size of the circular regions built on top of the spatial space. One direction would be testing this index structure on nearest neighbor queries and work on enhancing its structure if needed. Other direction would be to work on history and future queries since the current framework is developed to handle now queries. The promising thing in this approach is that the framework is easily extendable to support these queries since the publish/subscribe paradigm can incorporate any type of queries.

Due to limited resources and battery life of moving objects, we intend in future work to move the calculation of the half distance in a safe region from the moving object to the server side. By incorporating future queries with this approach, our framework will replace pull queries to push queries where our framework will be able to predict the future location of the moving object based on speed and velocity and push new results to the client automatically. Finally, in term of practice, we had implemented our framework and spatial index structure in SQL Server as temp tables in order to simulate the real behavior of spatial index. One future work would be to implement our index structure as part of the DBMS especially in an open source DBMS such as MySQL.

VII. REFERENCES

[1] Mohamed F. Mokbel, Xiaopeng Xiong, Moustafa A. Hammad, Walid G. Aref، "Continuous Query Processing of Spatio-temporal Data Streams in PLACE", Geoinformatica V.9, pp.343-365, 2005. doi: 10.1007/s10707-005-4576-7

[2] Z.W. Yuan, M. Cheng, H.S. Kim, H.Y. Bae and J. W. Ge, "Querying Spatial-temporal Data in Location-Based Services", 2005.

[3] Mohamed F. Mokbel, "Continuous Query Processing in Spatio-temporal Databases", Springerlink, p 364-367, 2005. doi: 10.1007/978-3-540-30192-9_10

[4] Mohamed F. Mokbel, Walid G. Aref, Susanne E. Hambrusch, Sunil Prabhakar, "Towards Scalable Locationaware Services: Requirements and Research Issues", GIS '03 Proceedings of the 11th ACM international symposium on Advances in geographic information systems, pp.110-117, 2003. doi: 10.1145/956676.956691

[5] D. Stojanov and S. Dordevic–Kajan, “Modeling and Querying Mobile Objects in Location–based Services”, FACTA UNIVERSITATIS (NI¡S), Ser. Math. Inform. 18 (2003), 59–80.

[6] Comarch technology, Technology Review (Comarch no.2), Poland, 2009.

[7] M. Koubaraki, Y. Theodoridis and T. Sellis, “Spatio-temporal Databases in the Years Ahead”, Spatio-Temporal Databases, Lecture Notes in Computer Science Volume 2520, pp.345-347, 2003. doi: 10.1007/978-3-540-45081-8_9

[8] Jun Zhang, Manli Zhu, Dimitris Papadias, Yufei Tao, Dik Lun Lee, “Location-based Spatial Queries”, In Proceedings of the 2003 ACM SIGMOD international conference on Management of data (SIGMOD '03). ACM, pp.443-454, 2003. doi: 10.1145/872757.872812

[9] Blur Marketing (2011, October), http://blur-marketing.com/blog/trends-and-statistics-in-location-based-services/

[10] Baihua Zheng, Dik Lun Lee, “Semantic Caching in Location-Dependent Query Processing”, Springer, Advances in Spatial and Temporal Databases, 97-113, 2001. doi: 10.1007/3-540-47724-1_6

[11] Dik Lun Lee, Manli Zhu, Haibo Hu, “When Location-Based Services Meet Databases”, ACM, Mobile Information Systems, vol.1, issue 2, pp.81-90, 2005.

[12] Z. Yuan, M. Cheng, H. Kim, H. Bae and J. Ge, Querying Spatial-temporal Data in Location-Based Services, 2005.

[13] Mohamed F. Mokbel, Xiaopeing Xiong, Walid G. Aref, “SINA: Scalable Incremental Processing of Continuous Queries in Spatio-temporal Databases”, In Proceedings of the 2004 ACM SIGMOD international conference on Management of data (SIGMOD '04), ACM, pp.623-634, 2004. doi: 10.1145/1007568.1007638

[14] Xiaopeng Xiong, Mohamed F. Mokbel, Walid G. Aref, S.E. Hambrusch, S. Prabhakar, “Scalable spatio-temporal continuous query processing for location-aware services”, IEEE, Proceedings 16th International Conference on Scientific and Statistical Database Management, pp.317-326, 2004. doi: 10.1109/SSDM.2004.1311223

[15] Shengsheng Wang, Chen Zhang, “A Dynamic Interval Based Circular Safe Region Algorithm for Continuous Queries in Moving Objects”, International Journal of Communications, Network and System Sciences, 2011. doi: 10.4236/ijcns.2011.45036

[16] Jingren Zhou, Per-åke Larson, Jonathan Goldstein, Luping Ding, “Dynamic Materialized Views”, IEEE, 23rd International Conference on Data Engineering, 2007. doi: 10.1109/ICDE.2007.367898.

How to cite

Ishraq Al-Fataftah, Jalal Atoum, "An Enhanced Framework for Improving Spatio-Temporal Queries for Global Positioning Systems". *International Journal of Research in Computer Science*, 3 (2): pp. 11-21, March 2013. doi: 10.7815/ijorcs. 32.2013.061